Técnicas Avanzadas de Programación

Practica Clase 2 – Diseño y Análisis de Algoritmos - Notación Big O

Objetivo general: Desarrollar la capacidad de diseñar e implementar algoritmos básicos propios, estimar su eficiencia mediante notación Big O y reflexionar sobre posibles mejoras.

Parte 1: Diseño e implementación de algoritmos Diseñar y programar en Python los siguientes cuatro algoritmos, cada uno en una sección diferenciada:

a) Encontrar el número más frecuente en una lista.
b) Verificar si una lista de enteros es palíndroma.
c) Contar la cantidad de elementos únicos en una lista.
d) Determinar si dos listas contienen los mismos elementos (sin importar el orden ni las repeticiones).

Para cada algoritmo:

- Pseudocódigo (en estilo PSeInt o similar).
- Estrategia: explicación de la lógica empleada.
- Implementación en Python usando buenas prácticas (nombres significativos, funciones, comentarios).
- Ejemplo de ejecución con entrada y salida esperada.
- Análisis de eficiencia: estimar la complejidad temporal con notación Big O y justificarla.
- (Opcional) Mejora del algoritmo: proponer una optimización y compararla con la versión original.

Parte 2: Reflexión

- ¿Cuál de los algoritmos fue más eficiente y por qué?
- ¿Qué dificultades surgieron al estimar la complejidad Big O?
- ¿Cómo ayuda este tipo de análisis en la toma de decisiones de programación?

Forma de entrega:

- Archivos: `.py` con los algoritmos y `.pdf` con pseudocódigo, explicaciones, ejemplos y análisis.
- Nombre de archivo: `clase2_Diseno_ApellidoNombre.pdf` y `clase2_Codigo_ApellidoNombre.py`.
- Plataforma de entrega: Moodle o según indicación del docente.

Criterio	Puntos
Correctitud de los algoritmos (4)	4 pts
Claridad del pseudocódigo y la implementación	2 pts
Análisis de eficiencia (Big O)	2 pts
Ejemplos de ejecución y explicaciones	2 pts